软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(3):582−593 [doi: 10.3724/SP.J.1001.2012.04003]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

XML 动态区间编码方法*

庄灿伟, 冯少荣+, 林子雨, 张东站

Dynamic Containment Labeling Scheme for XML

ZHUANG Can-Wei, FENG Shao-Rong+, LIN Zi-Yu, ZHANG Dong-Zhan

(Department of Computer Science, Xiamen University, Xiamen 361005, China)

+ Corresponding author: E-mail: shaorong@xmu.edu.cn

Zhuang CW, Feng SR, Lin ZY, Zhang DZ. Dynamic containment labeling scheme for XML. *Journal of Software*, 2012,23(3):582−593. http://www.jos.org.cn/1000-9825/4003.htm

Abstract: A novel containment scheme called DCLS is proposed to effectively process updates in dynamic XML data. DCLS generalizes the static containment scheme from integer order to vector order and thus completely avoids re-labeling when XML data updating. Moreover, DCLS is compact and efficient regardless of whether the documents are updated or not. On the one hand, DCLS uses integer-based static containment scheme for initial labeling, which yields compact size and excellent query efficiency for static documents. On the other hand, DCLS takes the integer as special vector, which not only deals with the case of document updating, but also achieves high query performance. Most importantly, DCLS can effectively avoid the rapid increase of labeling size for the case of skewed insertions. Experimental results confirm the benefits of this approach compared to previous dynamic containment schemes.

Key words: XML; document updating; labeling scheme; dynamic containment scheme; vector order

摘 要: 提出了适用于 XML 文档更新环境下的区间编码方法——DCLS(dynamic containment labeling scheme).DCLS 将基于整数的编码泛化到基于向量的编码,扩展了传统静态区间编码方法,有效避免了 XML 文档更新时的重新编码.不论文档更新与否,DCLS 都显示了良好的性能:DCLS 利用基于整数的静态区间编码方法进行初始编码,在文档不更新的环境下,具有较高的存储效率和查询性能;同时,DCLS 将整数视为特殊向量,不仅能够支持文档更新,而且更新效率高;特别是倾斜插入时,DCLS 可以避免编码位长的快速增加.实验结果表明,与已有的动态区间编码方法相比,DCLS 具有更好的性能.

关键词: 可扩展标记语言;文档更新;编码技术;动态区间编码;向量序
中图法分类号: TP311 文献标识码: A

随着网络应用的快速发展,XML(extensible markup language)数据正成为主流的数据形式.如何对 XML 数据建立有效索引以及实现高效查询,是当前的研究热点之一.大部分 XML 相关索引和查询技术基于某种对 XML 树的编码方法,其中最为流行的一类是区间编码.

* 基金项目: 国家自然科学基金(50604012), 中央高校基本科研业务费专项资金(2011121049)
收稿时间: 2009-12-31; 修改时间: 2010-07-28; 定稿时间: 2011-02-17

传统的静态区间编码方法[1]中,每个节点都被赋予一对整数,这对整数表达了节点覆盖区域,进而支持了节点间位置关系和结构关系计算.但是使用静态区间编码不能有效处理 XML 文档更新,一旦更新发生,整个树需要重新编码,系统代价高.为解决该问题,一些研究人员提出了动态区间编码方法,包括浮点数区间[2]、CDBS[3,4]以及 QED[5,6]等.相比较静态区间编码,这些方法支持文档更新操作,但同时需要更多时空开销,也降低了查询性能.特别在文档不更新或者少更新环境下,效率偏低.

静态区间编码和动态区间编码各有利弊.当文档不更新或者少更新时,静态区间编码无疑是更好的选择;但当 XML 频繁更新时,静态区间编码性能急剧下降,而动态区间编码则显示出优势.通常情况下,人们很难事先判断文档更新频率,进而不易选择合适的编码方法.这个意义下,开发出有效的动态编码,满足文档更新与否情况下都具有良好性能显得尤为重要.基于此,本文提出了新的动态区间编码方法——DCLS(dynamic containment labeling scheme),DCLS 直接在静态区间编码基础上进行扩展,可以有效地支持 XML 动态更新,同时又确保了文档不更新环境下的良好性能.本文贡献如下:

- 基于 Dewey 编码和 Dewey 序,阐述了向量支持 XML 更新的原理;
- 提出了基于向量的动态区间编码方法——DCLS.DCLS 将整数视为特殊向量,扩展了静态区间编码,可以支持 XML 动态更新;
- 设计了向量存储格式,只需按位比较就可以进行向量间偏序关系判断;
- 设计了实验,通过多个指标显示 DCLS 编码的有效性.

本文第 1 节介绍向量序支持 XML 更新的原理.第 2 节阐述利用向量进行区间编码的方法——DCLS.第 3 节给出向量的存储格式.第 4 节分析实验结果.第 5 节介绍相关工作.第 6 节总结全文.

1 向量和向量序

本节介绍 DCLS 的核心和基础——向量序.向量序保证了任意两个向量间可以无限插入新向量,从而支持 XML 动态更新.向量序的思想来源于 Dewey 编码[7],首先介绍 Dewey 编码,在此基础上介绍向量序及其支持中间插入的原理,接着给出中间向量计算算法.

1.1 Dewey编码和向量

Dewey 编码是一种基于路径的编码方法,每个节点的编码以父节点编码为前缀.给定一个节点 v,则其第 i 个孩子节点 u_i 的编码为 $L(u_i)=L(v).i$,其中,$L(a)$表示节点 a 的 Dewey 编码.为有效进行节点间在文档中位置关系的判断,引入了 Dewey 序.两个 Dewey 编码利用 Dewey 序进行偏序关系判断,Dewey 序定义如下:

定义 1(Dewey 偏序关系$\prec$).给定两个节点的 Dewey 编码 $A{:}a_1.a_2.....a_m$ 和编码 $B{:}b_1.b_2.....b_n$,$A\prec B$ 当且仅当满足下列两个条件之一:

- 条件 1:A 是 B 的祖先节点,即 $m<n$ 且 $a_1=b_1,a_2=b_2,...,a_m=b_m$;
- 条件 2:存在 $k<=\min(m,n)$,满足 $a_1=b_1,a_2=b_2,...,a_{k-1}=b_{k-1}$ 且 $a_k<b_k$.

利用 Dewey 序可以很好地进行文档树中节点间的位置关系判定.给定两个节点的 Dewey 编码 A 和 B,节点 A 位于节点 B 之前$\Leftrightarrow A\prec B$. □

基于 Dewey 序,可以得到以下定理:

定理 1. 给定 Dewey 编码 $A{:}a_1.a_2...a_m$ 和 $B{:}b_1.b_2...b_n$,其中 $A<B$,则存在新的 Dewey 编码 C,满足 $A\prec C\prec B$.

证明:当 $m\geq n$,令 $C=a_1.a_2...a_m.1$,则由 Dewey 偏序关系的判断条件 1 得 $A\prec C$,又由判断条件 2 得 $C\prec B$,所以 $A\prec C\prec B$;当 $m<n$,令 $C=a_1.a_2\ldots a_m.b_{m+1}-1$,则由判断条件 1 得 $A\prec C$,又由判断条件 2 得 $C\prec B$,所以 $A\prec C\prec B$. □

定理 1 是本文设计动态区间编码的理论基础,即在 Dewey 序关系下,给定有序的 Dewey 码序列,可以往任意位置插入新的 Dewey 码,使得新的序列仍然有序.

例 1:图 1 是 Dewey 编码示例,基于 Dewey 序,可以得到 12 个 Dewey 编码间的偏序关系如下:“1”$\prec$“1.1”$\prec$“1.2”$\prec$“1.2.1”$\prec$“1.2.2”$\prec$“1.3”$\prec$“2”$\prec$“2.1”$\prec$“2.2”$\prec$“2.2.1”$\prec$“2.2.2”$\prec$“2.3”.特别地,介于编码“1”和“2”之间的 Dewey

编码有“1.1”,“1.2”,“1.3”,....同理得到,任意两个 Dewey 编码间都可以找到介于其间的新的 Dewey 编码.

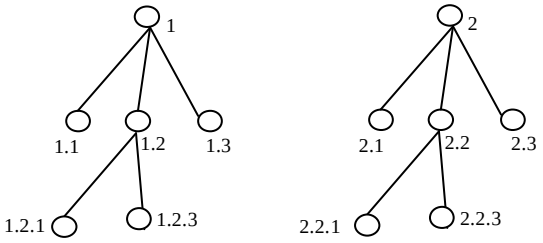


Fig.1 An example of Dewey labeling scheme

图 1 Dewey 编码示例

每个 Dewey 编码是一个向量,为叙述方便,我们用向量代指 Dewey 编码,并用向量序代指 Dewey 序.向量间利用向量序判断偏序关系.值得注意的是,整数是特殊向量.该性质使 DCLS 可以利用整数进行 XML 初始编码.

1.2 计算中间向量

XML 更新发生时,核心是计算两个向量间新的中间向量.定理 1 指出了中间向量的存在性和无穷性.为降低编码位长并提高查询性能,中间向量应具有较短位长.给定任意两个向量 A:$a_1.a_2...a_m$ 和 B:$b_1.b_2...b_n$,其中 $A\prec B$,目标是找到新的中间向量 C,满足 $A\prec C\prec B$,且 C 的位长尽可能短.以下阐述 C 的计算方法.

由向量序定义得知,A 与 B 满足序关系判断中的条件 1 或者条件 2.

(1) A 与 B 满足条件 1,即 $m<n$ 且 $a_1=b_1,a_2=b_2,...,a_m=b_m$ 时,令 $C=a_1.a_2...a_m.b_{m+1}-1$,则有 $A\prec C\prec B$(case 1);

(2) A 与 B 满足条件 2,即存在 $k<=\min(m,n)$,满足 $a_1=b_1,a_2=b_2,...,a_{k-1}=b_{k-1}$ 且 $a_k<b_k$,又分 3 种情况:

- 当 $n>k$ 或者 $b_k>a_k+1$,令 $C=a_1.a_2...a_{k-1}.(a_k+1)$,则有 $A\prec C\prec B$(case 2);
- 当 $m=n=k$ 且 $b_k=a_k+1$,令 $C=a_1.a_2...a_{k-1}.a_k.0$,则有 $A\prec C\prec B$(case 3);
- 当 $k=n<m$ 且 $b_k=a_k+1$,令 $C=a_1.a_2...a_{k-1}.a_k.a_{k+1}+1$,则有 $A\prec C\prec B$(case 4).

按以上规则得到的中间向量具有较短位长,其实现过程见算法 1.

算法 1. *GetMiddleVector*(A,B).

输入:向量 A:$a_1.a_2...a_m$ 和向量 B:$b_1.b_2.....b_n$,其中,$A\prec B$;

输出:向量 C,满足 $A\prec C\prec B$.

```
1.  k←1;
2.  while a_k= b_k do
3.      k++;
4.  endwhile                                  //向量 A 和 B 的第 k 维分量不同
5.  if k= m+ 1 then C←a_1.a_2.....a_m.b_{m+1}−1;        //case 1:k>m
6.  else if n>k||b_k>a_k+ 1 then C←a_1.a_2.....a_{k−1}.(a_k+ 1);   //case 2:k<= m && (k<n||b_k>a_k+ 1)
7.  else if m= k then C←a_1.a_2.....a_{k−1}.a_k.0;      //case 3:k= m= n && b_k= a_k+ 1
8.  else C←a_1.a_2.....a_{k−1}.a_k.a_{k+1}+ 1;          //case 4:k= n<m && b_k= a_k+ 1
9.  Endif
10. return C;
```

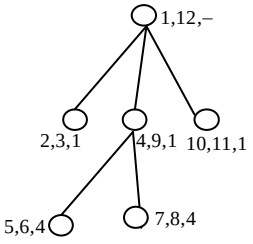
例 2:设 A=“1”,B=“1.0.0”,则有 C=“1.−1”(case 1);设 A=“1.−1”,B=“1.0.0”,则有 C=“1.(−1+ 1)”=“1.0”(case 2);设 A=“1”,B=“2”,则有 C=“1.0”(case 3);设 A=“1.−1.0”,B=“1.0”,则有 C=“1.−1.(0+ 1)”=“1.−1.1”(case 4).

2 DCLS:XML 动态区间编码机制

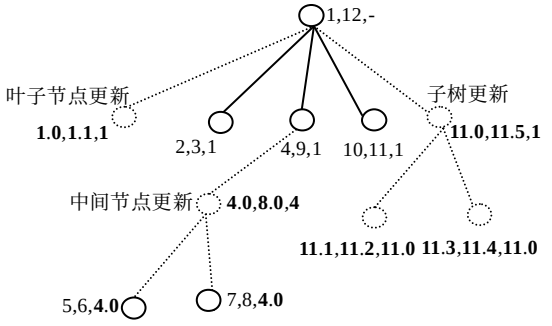
DCLS 利用向量进行区间编码,支持 XML 动态更新.本节介绍 DCLS 区间编码方法.

2.1 初始编码

DCLS 利用传统静态区间编码方法进行初始编码.为支持节点间兄弟关系判断,DCLS 采用了带父节点信息的区间编码 *P-Containment*[4],即节点 *u* 的编码 *L*(*u*)是三元组(*start*,*end*,*pstart*),其中,*start*,*end* 代表节点标签在文档出现的开始位置和结束位置,*pstart* 代表节点的父节点 *start* 值.图 2(a)是利用 DCLS 初始编码实例.



(a) 初始编码



(b) 叶子节点更新,中间节点更新和子树更新

Fig.2 An example of DCLS

图 2 DCLS 编码实例

DCLS 支持节点间关系判断,规则如下:

- 位置关系(Doc)判断:节点 *u* 位于节点 *v* 之前⇔*L*(*u*).*start*≺*L*(*v*).*start*;
- 祖先后代关系(AD)判断:节点 *u* 是节点 *v* 的祖先节点⇔*L*(*u*).*start*≺*L*(*v*).*start* 且 *L*(*u*).*end*≺*L*(*v*).*end*;
- 父子关系(PC)判断:节点 *u* 是节点 *v* 的父亲节点⇔*L*(*u*).*start*= *L*(*v*).*pstart*;
- 兄弟关系(Sibling)判断:节点 *u* 和节点 *v* 是兄弟节点⇔*L*(*u*).*pstart*= *L*(*v*).*pstart*.

2.2 动态更新支持

DCLS 支持 XML 叶子节点更新、中间节点更新和子树更新.

- 叶子节点更新

删除叶子节点不用涉及节点编码改变.当插入叶子节点 *u* 时,首先计算待插入节点满足区间(*left*,*right*):设 *u* 的左兄弟节点为 *ls*,右兄弟节点为 *rs*,父节点为 *p*.如果 *u* 的左兄弟节点存在,则 *left*= *ls.end*,否则 *left*= *p.start*;如果 *u* 的右兄弟节点存在,则 *right*= *rs.start*,否则 *right*= *p.end*.接着计算叶子节点 *u* 的编码:*u.start*= *GetMiddleVector*(*left*, *right*),*u.end*= *GetMiddleVector*(*u.start*,*right*),*u.parent*= *p.start*.图 2(b)显示了叶子节点更新实例,图中黑体部分表示更新操作中涉及改变的编码,下同.

- 中间节点更新

删除和插入中间节点都需要改变部分节点编码.删除中间节点时,需要改变被删节点所有孩子节点 *pstart* 域,因为它们的父节点发生改变.插入中间节点 *u* 时,设 *u* 父节点为 *p*,*u* 的 *n* 个孩子节点为 $c_1,c_2,...,c_n$,则 *u.start*= *GetMiddleVector*(*p.start*,c_1.*start*),*u.end*= *GetMiddleVector*(c_n.*end*,*p.end*);*u.pstart*= *p.start*.另外,需要改变 *u* 的各个孩子节点的 *pstart* 域,即 c_i.*pstart*= *u.start*(*i*= 1,2,...,*n*).图 2(b)显示了中间节点更新实例.

- 子树更新

删除子树不用涉及编码改变,讨论插入子树的情况.插入子树可以转化为单一叶子节点多次插入,但这样处理会使向量维数增加较快,所以应对插入的子树统一编码.首先找到待插入子树满足的区间(*left*,*right*),获得

left,*right* 方法和插入叶子节点一样.接着对子树进行深度优先遍历,计算节点 *start* 和 *end*.另外,利用父节点信息计算 *pstart*.在遍历过程中,假设子树共有 *n* 个节点,则共需产生 2*n* 个介于 *left* 和 *right* 之间的新向量,并赋予各个节点的 *start* 域和 *end* 域,这 2*n* 个新向量通过如下方法获得:

V_1= *GetMiddleVector*(*left*,*right*),V_2= *GetMiddleVector*(V_1,*right*),...,V_{2n}= *GetMiddleVector*(V_{2n-1},*right*).

图 2(b)显示了子树更新实例.

3 向量存储格式设计

关于 Dewey 编码的存储格式已有相关研究,包括 UTF-8[7],Ordpath[8]等,它们都可以直接应用到 DCLS 中.鉴于 DCLS 自身的特点,本文设计了新的向量存储格式.

3.1 DCLS格式

初始编码时,DCLS 采用一维向量进行编码,为有效降低位长,向量第一维采用定长存储.当动态更新时,为避免溢出产生,从向量第 2 维开始,采用变长存储方式.另外,由于每个向量维度不定,为表示一个向量的结束,在向量尾部连接上结束标志“00”.即向量 $a_1.a_2...a_n$ 的 DCLS 存储格式为:$a_1 \oplus a_2 \oplus ... \oplus a_n \oplus 00$,其中,$a_1$ 采用定长存储,而 $a_2,a_3,...,a_n$ 采用变长存储(“⊕”表示字符串连接操作,下同).

定长存储 a_1 相对简单,设初始 XML 文档的节点数为 *n*,则将 a_1 长度固定为 $1+\log_2(2n)$,此时可有效表达 1,2,...,2*n*.

$a_2,a_3,...,a_n$ 的存储见表 1,每个分量的基本格式为(长度域 *L*⊕值域 *V*),其中,长度域 *L* 指示 *V* 的位长.考虑到方便解码,*L* 的设置满足非前缀特性.另外,*V* 的位长 *length* 由 *L* 的位长 *k* 决定:当 *k*=2 时,*length*=0;当 *k*=3 时,*length*=1;当 *k*≥4 时,*length*=2×(*k*–3).表 2 是部分整数的(*L*⊕*V*)格式示例.

Table 1 (*L*⊕*V*) format of integer

表 1 整数的(*L*⊕*V*)格式

长度域 *L*	值域 *V* 的位长	整数表示范围
...	...	...
010000001	12	[–5462,–1367]
01000001	10	[–1366,–343]
0100001	8	[–342,–87]
010001	6	[–86,–23]
01001	4	[–22,–7]
0101	2	[–6,–3]
011	1	[–2,–1]
10	0	[0,0]
110	1	[1,2]
1110	2	[3,6]
11110	4	[7,22]
111110	6	[23,86]
1111110	8	[87,342]
11111110	10	[343,1366]
1111111110	12	[1367,5462]
...	...	...

Table 2 Examples of (*L*⊕*V*) format

表 2 (*L*⊕*V*)格式示例

十进制整数	(*L*⊕*V*)格式
...	...
–7	01001,1111
–6	0101,00
–5	0101,01
–4	0101,10
–3	0101,11
–2	011,0
–1	011,1
0	10
1	110,0
2	110,1
3	1110,00
4	1110,01
5	1110,10
6	1110,11
7	11110,0000
...	...

由算法 1 可知,当 XML 进行动态更新时,涉及到整数的加 1 和减 1 计算,因此我们设计算法实现(*L*⊕*V*)格式下整数的加 1 操作 *Self-Increase* 和减 1 操作 *Self-Decrease*,如算法 2 和算法 3 所示.

算法 2. *Self_Increase*(*S*).

输入:*L*⊕*V* 格式的整数 *S*;

输出:*L*⊕*V* 格式的整数 *S*+1.

```
1  k←1;                                   //第 1 行~第 6 行计算长度域 L 的长度 k
2  if S[0]= '0'
3  then  k++; while S[k]= '0' do k++; endwhile
```

```
4   else while S[k]= '1' do k+ + ; endwhile
5   endif
6   k+ + ;
7   if k= 2 then lengt←0;                          //第 7 行~第 10 行计算值域的长度 length
8   else if k= 3 then length←1;
9   else length←2*(k–3);
10 endif
11 p←k+ length–1;                                  //第 11 行~第 13 行计算整数加 1 操作
12 while S[p]= '1' do    S[p]←'0';    p––; endwhile
13 S[p]←'1';
14 if (S[0]= '0' && (p= k–2)||p= 0 then            //第 14 行~第 20 行进行边界处理
15       if p<= 2 then S←S.erase(S.size()–2,2);
16       else S←S.erase(S.size()–3,3); endif
17 else if    S[0]= '1' && p= k–1 then
18       if p<= 2 then S←S.insert(S.size(),2,'0');
19       else S←S.insert(S.size(),3,'0'); endif
20 endif
21 return S;
```

算法 **3**. *Self_Decrease*(*S*).

输入:*L*⊕*V* 格式的整数 *S*;

输出:*L*⊕*V* 格式的整数 *S*–1.

```
1   k←1;                                           //第 1 行~第 6 行计算长度域 L 的长度 k
2   if S[0]= '0';
3   then    k+ + ; while S[k]= '0' do k+ + ; endwhile
4   else while S[k]= '1'; do k+ + ; endwhile
5   endif
6   k+ + ;
7   if k= 2 then lenght←0;                         //第 7 行~第 10 行计算值域的长度 length
8   else if k= 3 then length←1;
9   else length←2*(k–3);
10 endif
11 p←k+ length–1;                                  //第 11 行~第 13 行计算整数减 1 操作
12 while S[p]= '0' do    S[p]←'1'; p––;    endwhile
13 S[p]←'0';
14 if S[0]= '1' && p= k–2 then                     //第 14 行~第 20 行进行边界处理
15       if p<= 2 then S←S.erase(S.size()–2,2);
16       else S←S.erase(S.size()–3,3); endif
17 else if (S[0]= '0' && p= k–1)||p= 0 then
18       if p<= 2 then S←S.insert(S.size(),2,'1');
19       else S←S.insert(S.size(),3,'1'); endif
20 endif
21 return S;
```

算法 2 中,首先解析长度域并计算长度域的长度 k(第 1 行~第 6 行):当 S[0]为‘0’,则出现的第 2 个‘1’为长度域结束标志;当 S[0]为‘1’,则出现的第 1 个‘0’为长度域结束标志.接着,利用 k 计算值域 V 的实际长度 *length*(第 7 行~第 10 行).之后,计算整数的加一操作(第 11 行~第 13 行):设 S 中最后出现‘0’的位置为 p,则将 $S[p]$置为‘1’,且将随后的‘1’置为‘0’.另外,由于值域的长度并不固定,需要进行边界处理(第 14 行~第 20 行).

算法 3 与算法 2 类似.

算法 2 和算法 3 直接利用整数的物理表示进行加 1 和减 1 操作,不需要在物理表示和逻辑表示间进行转化,具有较好的时间效率.

3.2 DCLS格式

利用 DCLS 格式存储向量,只需按位比较(即字典序)就可以进行向量间偏序关系判断.首先引入两个引理.

引理 1. 设位串 S_1 和 S_2 表示定长存储下的整数 i_1 和 i_2,则 $i_1<i_2\Leftrightarrow S_1\prec S_2$.这里,字符串 S_1 和 S_2 的比较采用字典序,下同.

证明:定长存储下,整数序用存储位串的字典序判断,得证. □

引理 2. 设位串 S_1 和 S_2 表示($L\oplus V$)格式下的整数 i_1 和 i_2,则 $i_1<i_2\Leftrightarrow S_1\prec S_2$.

证明:设 $S_1=L_1\oplus V_1$,$S_2=L_2\oplus V_2$.充分性证明,由于 $i_1<i_2$,可得情况 1:$L_1\prec L_2$ 或者情况 2:$L_1=L_2$ 且 $V_1\prec V_2$.而这两种情况都使得 $S_1\prec S_2$,得证.逆推可证必要性.命题得证. □

利用引理 1 和引理 2 可以推导出向量间偏序关系的判断规则,如定理 2.

定理 2. 设位串 S_1 和 S_2 表示 DCLS 格式下的向量 A 和 B,则 $A\prec B\Leftrightarrow S_1\prec S_2$.

证明:设向量 A:$a_1.a_2.....a_m$ 和 B:$b_1.b_2.....b_n$ 的 DCLS 格式分别为 S_1:$a_1\oplus a_2\oplus...\oplus a_m\oplus 00$ 和 S_2:$b_1\oplus b_2\oplus...\oplus b_n\oplus 00$.充分性证明,当 A 是 B 的前缀,此时 $S_1=a_1\oplus a_2\oplus...\oplus a_m\oplus 00$,$S_2=a_1\oplus a_2\oplus...\oplus a_m\oplus b_{m+1}\oplus...\oplus b_n\oplus 00$,有 $S_1\prec S_2$.当 A 不是 B 前缀,即存在 $k\le\min(m,n)$,满足 $a_1=b_1,a_2=b_2,...,a_{k-1}=b_{k-1}$ 且 $a_k<b_k$,此时 $S_1=a_1\oplus a_2\oplus...\oplus a_{k-1}\oplus a_k\oplus a_{k+1}\oplus...\oplus a_m\oplus 00$,$S_2=a_1\oplus a_2\oplus...\oplus a_{k-1}\oplus b_k\oplus b_{k+1}\oplus b_n\oplus 00$.由引理 1 和引理 2 得 $S_1\prec S_2$.逆推可证必要性. □

例 3:设第 1 维位数设定为 4,则向量“1.–1.2”的 DCLS 格式为“0001 0111 1101 00”,向量“1.0.1”的格式为“0001 10 1100 00”.同时,利用各唯分量的非前缀特性,也可以将 DCLS 格式“00010111110100”和“000110110000”解析回向量“1.–1.2”和“1.0.1”.

另外,由于按位比较“00010111110100”≺“00011100110000”,所以向量“1.–1.2”≺“1.0.1”.

4 实验设计与结果

选用 CDBS 和 QED 区间编码与本文提到的 DCLS 进行比较.在实现中,CDBS 采用定长长度域(位长为 6)的存储方法.另外,CDBS 和 QED 都采用能重用已删编码的动态更新算法[4,6].

实验选用 Tinyxml 解析 XML 文档,各种编码用 DEV-C++实现,实验平台为 1.8GHz Intel 酷睿双核处理器,2G 内存,Windows 7 操作系统.选用的测试数据集及属性见表 3,其中,$D1$,$D2$ 源自文献[9],$D3$~$D5$ 源自文献[10].

Table 3 Test datasets

表 3 测试数据集

数据集	文档名称	总结点数	最大深度	平均深度
D1	Hamlet	6 636	6	4.79
D2	All_shakes	179 690	7	5.58
D3	Nasa	476 646	8	3.16
D4	Lineitem	1 022 976	3	2.94
D5	Treebank	2 437 666	36	7.87

4.1 静态性能分析

实验用 CDBS,QED 和 DCLS 分别对上述 5 个数据集进行初始编码,并测试它们的静态性能.

- 静态编码时间.

图 3(a)是三者的编码时间对比,可以看出,DCLS 具有最好的时间性能.这是因为 CDBS 和 QED 在静态编码前需要递归生成所有的 CDBS/QED 码值序列,然后利用生成的码值序列对节点编码.而 DCLS 直接利用连续的自然数进行编码,不需要额外生成一个序列.另外,由于需要进行费时的除 3 操作,QED 需要最长的时间代价.

- 额外空间需求.

额外空间需求主要来自于编码过程中临时表空间的占用,实验只通过临时表大小衡量额外空间需求.图 3(b)是三者的额外空间需求对比.可以看出,随着节点数增加,CDBS 和 QED 表空间占用也增加,而 DCLS 不需要临时表.这是由于 CDBS 和 QED 需要先生成含有 $2n$ 个元素的码值表,额外空间复杂度为 $O(n)$.而 DCLS 利用连续的自然数进行编码,不需要额外码值表.当文档较大而内存有限时,将限制 CDBS 和 QED 的应用.

- 编码位长对比.

图 3(c)是三者的平均编码位长对比,从图中可以看出,QED 编码位长最长.这是因为 QED 利用四进制进行编码,且四进制中的"0"只用来作编码结束标志而不出现在有效位中,降低了 QED 存储比.与 CDBS 相比,DCLS 编码位长较短.这是因为 DCLS 的定长存储效率比 CDBS 的变长存储效率高.

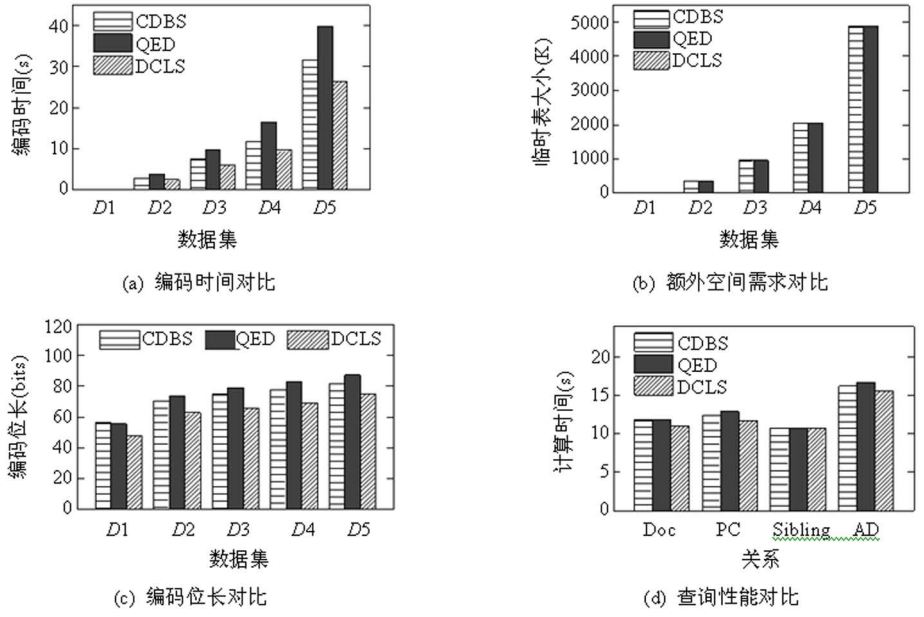


Fig.3 Performance study on initial labeling

图 3 静态编码性能对比

- 查询性能对比.

在 XML 查询中,4 种关系的计算是常见的,分别是节点间在文档中出现的先后顺序(doc)、两个节点是否具有父子关系(PC)、祖先后代关系(AD)或兄弟关系(sibling).实验选出文档 Treebank 的前 10 000 个节点并计算每两个节点间是否具有上述 4 种关系.图 3(d)显示了各种编码方法计算 4 种关系所需要的时间.由于 CDBS,QED 和 DCLS 都利用字典序进行偏序关系比较,因此节点编码长度是影响查询性能的主要因素.DCLS 由于具有最短的编码位长,使得 DCLS 查询性能最好.

4.2 动态性能分析

测试往两个相邻节点间进行均匀插入和倾斜插入两种情况下的动态性能.均匀插入和倾斜插入定义如下:

- 均匀插入:往两个相邻节点 *left*,*right* 间均匀插入 $2^n-1(n=1,2,3,...)$个节点.这 2^n-1 个节点插入方法如下:当 $n=1$ 时,在 *left*,*right* 间插入 1 个节点;当 $n=k+1(k=1,2,3,...)$时,在 $n=k$ 的基础上,在 *left*,*right* 间每两个相邻节点间再插入一个新节点,即新增加 2^k 节点;

- 倾斜插入:在同一个节点之前或之后连续插入 n 个节点.

4.2.1 均匀插入

- 编码位长

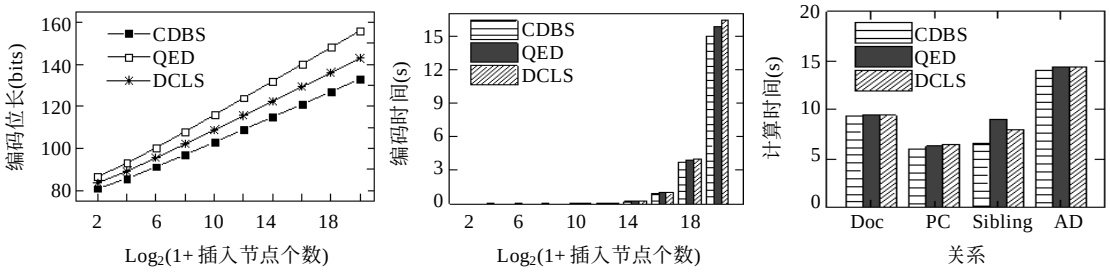
均匀插入 $2^n-1(n=1,2,3,...,20)$个节点时,三者的编码位长增长都是 $O(n)$.实验统计插入节点的实际编码位长对比,图 4(a)显示了 CDBS 位长增长最慢,DCLS 次之,而 QED 最快.CDBS 直接利用二进制进行编码,具有很好的存储比,而 QED 只利用了四进制中的“1”,“2”,“3”作为有效位进行编码,降低了存储效率.从图中可以看出,DCLS 的存储比介于 CDBS 和 QED 之间.

- 编码时间

计算新编码时,三者都需要对已有编码进行整串扫描,时间复杂度相同.图 4(b)是三者的实际编码时间对比,可以看出,与 QED 相比,CDBS 由于编码长度相对较短花费较少的时间开销,而 DCLS 时间开销最大.这主要是由于 DCLS 编码是一个多维向量,解码多维向量需要花费时间,但差异并不显著.

- 查询性能

往 Treebank 根下均匀插入 8 191(即 $n=13$)个新节点,并利用插入节点测试查询性能,图 4(c)是各种编码方法对所有插入节点两两之间计算前述 4 种关系所需要的时间对比.CDBS 具有最短的编码长度使其查询性能最好,而 DCLS 介于 CDBS 和 QED 之间.



(a) 编码位长对比 (b) 编码时间对比 (c) 查询性能对比

Fig.4 Performance study on uniform insertions

图 4 均匀插入性能对比

4.2.2 倾斜插入

倾斜插入时,QED 和 DCLS 都能有效地避免溢出,而 CDBS 却不能,因此我们只进行 QED 和 DCLS 性能对比.实验随机选择 Treebank 根下一个节点,测试往该节点之后倾斜插入 n 个新节点时 QED 和 DCLS 的动态性能.

- 编码位长

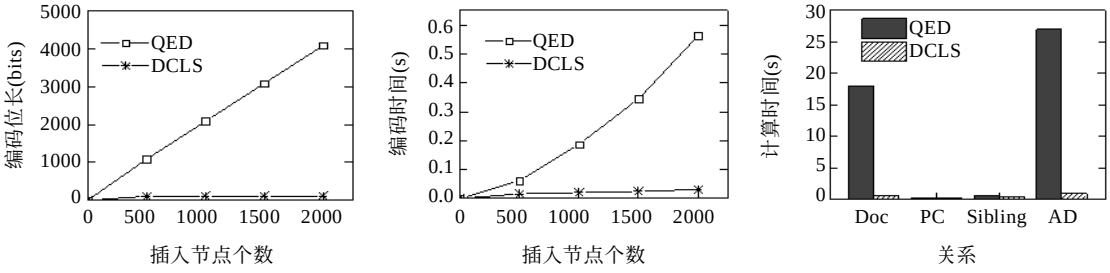
图 5(a)是插入节点的平均编码位长对比.可以看出,QED 编码位长迅速增加,而 DCLS 缓慢.当插入 2 000 个节点时,QED 的平均编码长度达到了 4 082 位而 DCLS 仅为 114 位.这是因为倾斜插入 n 个节点时,QED 编码位长增加 $O(n)$,而 DCLS 仅需 $O(\log n)$.

- 编码时间

图 5(b)是倾斜插入编码时间对比.当动态更新时,计算新编码的时间与编码长度相关.由于编码位长的显著差异,QED 与 DCLS 的编码时间也存在较大差异.

- 查询性能

往 Treebank 根下倾斜插入 2 000 个新节点,并对插入的节点进行查询性能测试.图 5(c)是各种编码方法对所有插入节点两两之间计算前述 4 种关系所需要的时间对比.可以看出,在计算 PC 和 Sibling 关系时,二者相差不大.这是由于采用了(*start*,*end*,*pstart*)格式下的区间编码,计算 PC 和 Sibling 时,只需与编码中的父节点信息 *pstart* 进行对比,而 *pstart* 在倾斜插入时保持不变,所以倾斜插入对 PC 和 Sibling 的关系计算影响不大.但当计算 Doc 和 AD 关系时,QED 由于编码位长较大,使得查询性能低下.



(a) 编码位长对比 (b) 编码时间对比 (c) 查询性能对比

Fig.5 Performance study on skewed insertions

图 5 倾斜插入性能对比

5 相关工作

早期的编码研究主要集中于静态 XML,Dewey 编码[7]和区间编码[1]是其中两种常用编码方法,二者都能高效地支持节点间结构关系查询.如需要判断两节点的祖先后代关系,Dewey 编码只需判断一节点的编码是不是另外一节点编码的前缀即可;区间编码则判断一个节点区间是否包含另一个节点区间.这两种编码都利用了整数对 XML 节点进行序关系标识,如 Dewey 编码用整数标识节点在所有兄弟节点中的顺序,区间编码编码用整数标识遍历顺序.利用整数标识顺序关系,具有序关系判断简单、处理高效、冗余度小、压缩比高等优点,但是不能支持 XML 更新,一旦更新发生,需要对大量节点重新编码,大大影响查询效率.

减少编码更新代价的一种方法是采用预留的策略[11],即在编码时预留编码空间,但当预留空间用完,仍不可避免重新编码.另外一种有效方法是利用新的偏序集而不是整数标识顺序,为支持 XML 动态更新,新的偏序集$\langle A,\prec\rangle$(A 是编码的集合,$\prec$是 A 上偏序关系)需满足:① 任意 $u,v\in A$,存在 $s\in A$,使得 $u\prec s\prec v$;② “$\prec$”是全序关系,即 A 上任意两个不同的编码都可以比较大小.基于这个思路,学者们提出了很多动态编码方法.浮点数区间编码[2]利用了任意两个浮点数间可以无限插入的性质,支持文档更新.但其缺点也是明显的,浮点数的机器表示位长有限,溢出问题不可避免;另外,浮点数频繁比较也会牺牲数据库查询性能.CDBS[3,4]和 QED[5,6]利用位字符串进行编码,并用字典序进行序关系判断.CDBS 码是以“1”结束的二进制字符串,任意两个以“1”结束的字符串之间存在无穷个满足字典序的字符串,使得 CDBS 支持更新.CDBS 有较高编码效率,但不可避免地产生溢出.QED 提出了基于四进制字符串编码的方法,四进制中“1”,“2”,“3”是有效编码,而“0”作编码结束标志.与 CDBS 相比,QED 避免了溢出,但需要更长编码位数,降低了查询性能.另外,倾斜插入时编码位长快速增加,也是 CDBS 和 QED 难于克服的缺点.OrdPath[8]是支持 XML 更新的前缀编码方法,其只利用奇数作顺序标志,而偶数作占位符使用,避免了重新编码,且对倾斜插入不敏感.但 OrdPath 码中同层节点编码段数不一致,降低了查询效率.DDE[12]利用多维向量加法的性质,将静态 Dewey 扩展成更新支持的编码方法.DDE 利用 Dewey 进行初始编码,具有很好的静态性能.当更新发生时,只需往相邻节点 $A{:}a_1.a_2...a_n$ 和 $B{:}b_1.b_2...b_n$ 插入新编码 $A+B{:}(a_1+b_1).(a_2+b_2)...(a_n+b_n)$即可,更新效率高.DDE 将序关系由一维的整数扩展到二维的分数,支持了文档更新.但是和整数序相比,新的序关系判断需要涉及向量解码和分数比较等操作,影响查询效率.另一方面,DDE 简单利用向量相加计算中间向量,不能对已删编码进行重用,编码长度较长.较长的编码除了占用更多空间外,也会降低查询性能.特别是删除和插入发生频繁时,编码长度增长更加明显.为有效利用已删节点编码,降低更新节点的编码长度,文献[13]提出了改进的中间向量计算算法,其产生的中间向量具有最短位长.

6 结束语

提出了一种新的高效的动态 XML 树编码方法——DCLS.DCLS 利用整数进行初始编码,具有良好的静态性能.同时,DCLS 利用向量进行动态编码,支持文档更新而且更新效率高;特别当倾斜插入 n 个节点时,DCLS 编

码长度仅增加 $O(\log n)$,克服了已有动态区间编码方法 $O(n)$的缺点.此外,DCLS 只需按位比较就可判断向量间偏序关系,具有良好的查询性能.

文献[14]指出当插入 n 个节点时,任何动态编码方法在不涉及改变原有编码的前提下,最坏情况编码长度增加 $O(n)$,DCLS 也不例外(每次新节点插入在之前插入的两节点间,虽然实际更新环境中很少出现这种情况).借鉴红黑树等经典动态树形结构的思想(最坏插入情况下仍能保持树高平衡),开发出更有效的动态编码方法,在允许改变少量已有编码的前提下,满足最坏情况编码长度只增加 $O(\log n)$,是编码更新问题未来的研究方向.

References:

[1] Zhang C, Naughton J, De Witt D, Luo Q, Lohman G. On supporting containment queries in relational database management systems. In: Aref WG, ed. Proc. of the 2001 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2001). Santa Barbara: ACM Press, 2001. 425−436. [doi: 10.1145/375663.375722]

[2] Amagasa T, Yoshikawa M, Uemura S. QRS: A robust numbering scheme for XML documents. In: Dayal U, Ramamritham K, Vijayaraman TM, eds. Proc. of the 19th Int'l Conf. on Data Engineering (ICDE 2003). Bangalore: IEEE Computer Society, 2003. 705−707. [doi: 10.1109/ICDE.2003.1260842]

[3] Li CQ, Ling TW, Hu M. Efficient processing of updates in dynamic XML data. In: Liu L, Reuter A, Whang KY, Zhang J, eds. Proc. of the 22nd Int'l Conf. on Data Engineering (ICDE 2006). Atlanta: IEEE Computer Society, 2006. 13−22. [doi: 10.1109/ICDE.2006.58]

[4] Li CQ, Ling TW, Hu M. Efficient updates in dynamic XML data: From binary string to quaternary string. VLDB Journal, 2008, 17(3):573−601. [doi: 10.1007/s00778-006-0021-2]

[5] Li CQ, Ling TW. QED: A novel quaternary encoding to completely avoid relabeling in XML updates. In: Grossman DA, Gravano L, Zhai C, Herzog O, Evans DA, eds. Proc. of the 14th ACM Int'l Conf. on Information and Knowledge Management (CIKM 2005). Bremen: ACM Press, 2005: 501−508. [doi: 10.1145/1099554.1099692]

[6] Li CQ, Ling TW, Hu M. Reuse or never reuse the deleted labels in XML query processing based on labeling schemes. In: Lee ML, Tan KL, Wuwongse V, eds. Proc. of the 11th Int'l Conf. on Database Systems for Advanced Applications (DASFAA 2006). LNCS 3882, Berlin: Springer-Verlag, 2006. 659−673. [doi: 10.1007/11733836_46]

[7] Tatarinov I, Viglas SD, Beyer K, Shanmugasundaram J, Shekita E, Zhang C. Storing and querying ordered XML using a relational database system. In: Franklin MJ, Moon B, Ailamaki A, eds. Proc. of the 2002 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2002). Madison: ACM Press, 2002. 204−215. [doi: 10.1145/564712.564715]

[8] O'Neil P, O'Neil E, Pal S, Cseri I, Schaller G, Westbury N. ORDPATHs: Insert-Friendly XML node labels. In: Weikum G, König AC, Deßloch S, eds. Proc. of the 2004 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2004). Paris: ACM Press, 2004. 903−908. [doi: 10.1145/1007568.1007686]

[9] NIAGARA experimental data. http://www.cs.wisc.edu/niagara/data

[10] University of Washington XML repository. http://www.cs.washington.edu/research/xmldatasets

[11] Luo DF, Meng XF, Jiang Y. Updating of extended preorder numbering scheme on XML. Journal of Software, 2005,16(5):810−818 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/810.htm [doi: 10.1360/jos160810]

[12] Xu L, Ling TW, Wu H, Bao ZF. DDE: From Dewey to a fully dynamic XML labeling scheme. In: Çetintemel U, Zdonik SB, Kossmann D, Tatbul N, eds. Proc. of the 2009 ACM SIGMOD Int'l Conf. on Management of Data (SIGMOD 2009). Providence: ACM Press, 2009. 719−730. [doi: 10.1145/1559845.1559921]

[13] Zhuang CW, Feng SR, Lin ZY, Zhang DZ. IDD: An Improved Method for DDE. Journal of Computer Research and Development,2010,47(suppl.):119−126 (in Chinese with English abstract).

[14] Cohen E, Kaplan H, Milo T. Labeling dynamic XML trees. In: Popa L, ed. Proc. of the 21st ACM Symp. on Principles of Database Systems (PODS 2002). Madison: ACM Press, 2002. 271−281. [doi:10.1145/543645.543648]

附中文参考文献:

[11] 罗道峰,孟小峰,蒋瑜.XML 数据扩展前序编码的更新方法.软件学报,2005,16(5):810−818. http://www.jos.org.cn/1000-9825/16/810.htm [doi: 10.1360/jos160810]

[13] 庄灿伟,冯少荣,林子雨,张东站.IDD:DDE 编码改进方法.计算机研究与发展,2010,47(增刊):119–126.

庄灿伟(1984－),男,福建泉州人,硕士生,主要研究领域为 XML 数据库.

林子雨(1978－),男,博士,助理教授,CCF 会员,主要研究领域为数据库,实时主动数据仓库,数据挖掘.

冯少荣(1964－),男,博士,副教授,CCF 会员,主要研究领域为 XML 数据库,数据仓库,数据挖掘,机器学习.

张东站(1974－),男,博士,副教授,CCF 会员,主要研究领域为数据库理论与技术.

第 9 届中国计算机图形学大会 Chinagraph 2012

2012 年 10 月 17—19 日 中国 成都

征文通知

由中国计算机学会、自动化学会、工程图学会、图像图形学会、系统仿真学会、香港多媒体及图像计算学会和国家自然科学基金委主办的第 9 届中国计算机图形学大会(Chinagraph ’2012)将于 2012 年 10 月 17—19 日在四川省成都市举行。

征稿范围(包括但不限于)

图形学基础理论与算法
真实感图形
几何造型与处理
计算机动画与游戏
基于图象和视频的图形技术
非真实感图形
基于图形硬件的通用计算技术
计算机辅助几何设计
计算机辅助设计
虚拟现实技术及应用
媒体计算
科学计算可视化
计算机仿真
人机交互技术
工程图形及应用
遥感及其相关技术

重要日期

征稿截止: 2012 年 5 月 6 日
录用通知: 2012 年 6 月 18 日
会议日期: 2012 年 10 月 17 日—19 日

联系方式

联系人:张严辞 刘艳丽 吴志红
联系电话:028-85412159
E-mail: chinagraph2012@gmail.com